SUMMARY OF TECO COMMANDS

Lower Case

a	without argument: appends next page to buffer with positive argument between 1 and 3: takes from 1 to 3 characters following the a, packs them left justified, and stands in expression as that value. with negative argument between -1 and -3: takes from 1 to 3 characters following current position of pointer in buffer, does as above. Pointer moves over characters examined.
b	has the value of the position at the beginning of the buffer.
c	with single argument, positive or negative: moves over the specified number of characters.
d	with single argument, positive or negative: deletes specified number of characters after pointer (before pointer if arg is negative)
e	inserts stop code at pointer.
f	with single positive argument: feed specified lines of blank tape
g	with single argument consisting of single q register into which block of characters has been 'x'ed: inserts previously stored block of characters at pointer. with no argument: does garbage collection.
h	with single positive argument: inserts specified number of characters following the h into buffer at pointer.
i	with no argument and sw3 down: inserts into buffer at pointer characters following 'i' until center dot. with no argument and sw3 up: takes character immediately following 'i' as delimiter; inserts characters until second occurance of delimiter. with argument magnitude less than 4096: inserts argument into buffer at pointer as signed decimal number. with argument magnitude greater than 4096: inserts argument into buffer at pointer as concise code, trailing spaces eliminated.

j	with single argument which is position in buffer: moves pointer to specified position.
k	with single positive argument: deletes characters from pointer to and including nth following carriage return. with single negative argument: deletes characters from but not including nth previous carriage return to pointer. with double argument: deletes characters between specified two buffer positions.
l	with single positive (negative) argument: moves pointer forward (backward) until nth following (preceeding) carriage return. Pointer is left following the carriage return.
m	with no argument: inserts commands from typewriter in command string following the 'm'; commences executing same when upper case followed by lower case is typed in. with single argument: defines macro having as its name the characters between the 'm' and the first following comma (only first three are recognized). Macro definition consists of characters between name-terminating comma and first occurance of overbar immediately followed by period. Nested definitions are not allowed. Macros are called by placing an overbar immediately before the name, and a comma immediately after the name. Nested calls are allowed. Macros may be redefined.
n	causes the contents of the buffer to be punched and the program to halt. 'CONTINUE' to execute additional commands.
o	with single positive argument: inserts specified number of characters into buffer at pointer. Characters are indicated by their concise code values typed in octal following the 'o'.
p	with single positive argument: punches contents of buffer and reads in next page specified number of times. Does not attempt to read in new page after last page punched if 'w' immediately follows 'p'.
q	followed by single letter or number: designates one of 36 available temporary storage registers.
r	with single argument, positive or negative: moves pointer in reverse over specified number of characters.

s	with no argument: presumes string following 's' in format same as 'i' (including sw3 option). Moves pointer forward until identical string is found. If string is not found in buffer, contents of buffer are punched and new page is read in and search continues. If sw6 is up, search is discontinued if string is not in buffer; string is typed out followed by question mark; commands following 's' are ignored. with single argument which is position of block of characters including terminator as specified above located in buffer or 'q' storage: searches for string in storage instead of following 's'. sw6 option as above.
t	takes arguments in same forms as 'k'. Types out characters on on-line typewriter. Pointer does not move.
u	stores argument in 'q' register immediately following 'u'.
v	same as 't' except displays characters on scope instead of typing.
w	prevents 'p' from reading in new page after last page has been punched.
x	takes arguments in same form as 'k'; must be followed by 'q' register. Places characters in storage. Leaves address in specified 'q' register such that characters may be retrieved by 'g'. Pointer does not move; characters are not deleted from buffer.
y	deletes entire contents of buffer, reads in new page.
z	has the value of the position at the end of the buffer.

Upper Case

A	causes longitudinal parity character to be punched following stop code for all succeeding pages. Check character consists of xor of all characters on page with delete bit punched. TECO always checks line of tape immediately following stop code. If it does not have the delete bit punched, it is ignored. If the delete bit is punched, the character is compared with one computed while reading the page, and a disagreement causes an informative typeout.

B	with single positive argument: causes output to be written on specified tape unit. with single negative argument: causes output to be written on drum field number as specified by magnitude of argument.
C	causes the entire stored command string to be inserted in the buffer at the pointer. Command string is not changed.
D	with single argument: sets up size of characters for display routine. 2 is minimum displayable size. Routine is initially set to 4.
E	causes TECO to discontinue punching longitudinal parity check character (A) for succeeding pages.
F	with arguments same as 's': operates same as 's' except does not punch pages.
G	kills present command string and buffer and commences reading command tape.
H	dismisses reader and punch (time share).
I	with single argument: specifies number of places between tab stops for display.
J	unused
K	kills entire command string, moves entire buffer to command string, kills entire buffer, commences executing commands at beginning of string just moved. All macro definitions are killed. Loop depth is set to 0.
L	with no argument: rewinds both tape units being used. with single positive argument: rewinds specified tape unit.
M	unused
N	unused
O	with single argument: causes argument to be printed on line as signed octal number.
P	causes output to be punched, clears punch buffer, requests punch. If punch is not available, 'punch busy' is typed. Typing in carriage return causes request to be made again.
Q	unused

R	causes input to be taken from paper tape reader, clears reader buffer, requests reader. If reader unavailable, action as in 'P'.
S	with arguments same as 's': operates same as 's' except does not move beyond end of current buffer if string is not found. If string is found, entire 'S' commands has value -1. If string is not found, pointer is at end of buffer and entire 'S' command has value 0.
T	with single positive argument which is tape unit number: causes input to be taken from specified tape unit instead of paper tape reader. with single negative argument: causes input to be taken from drum field number as specified by magnitude of argument.
U	unused
V	has the value of the number of carriage returns passed over during the last search ('s', 'F' or 'S').
W	has the value of the test word.
X	unused
Y	unused
Z	unused

Other Characters

,	separates arguments for double argument commands; terminates macro names; indicates point at which value for loop control is found.
=	with single argument: causes argument to be printed on line as signed decimal number.
.	has value of pointer.
×	indicates multiplication in command string expression.
/	indicates division if within command string expression; indicates start of comment if not within expression. Comment continues until next carriage return. Comment is typed on line if sw4 is up or if first / is followed by a second.
.	terminates search and insert strings if sw3 down.

-	indicates subtraction in command string expression.
‾	indicates beginning of macro name if followed by legal macro name character (letter or number), indicates end of macro definition if followed by period.
{	[center dot, open paren] indicates start of loop. Loop is executed if expression immediately before next comma is negative. If expression is zero or positive, commands are skipped until matching } [center dot, close paren] is encountered.
}	[center dot, close paren] loops back to matching { [center dot, open paren].

ADDITIONS AND CORRECTIONS TO <u>TECO</u> MEMO

Zero or negative arguments on 'l' (line) commands now work.

There are now no circumstances under which a search will fail if the designated string of characters is actually present.

If two arguments, which are buffer positions, are given for the 'k' (kill) command, all material between the two positions will be deleted. If one argument is given, it works as before, i.e., the argument is taken to be a number of lines to be excised.

Examples:

5k	deletes from current position of pointer up to and including the fifth following carriage return.
.-5,.k	deletes five characters immediately before current position of the pointer.
b,zk	deletes entire page except for the last stop code

when two arguments are given, the prior position of the pointer has no effect (unless, as in example two above, '.' (point) is a constituent of one or both arguments). After the execution of the commands, the pointer will be at the point where the material was removed.

The 't' (type) command may be given two arguments which are positions in which case, all material between the two positions will be typed out. The pointer does not move.

Examples:

3t	types from current position of pointer up to and including the third following carriage return
.-5,.+2t	types seven characters, 5 before and two after current position of pointer
b,zt	types entire page

Plus sign (+) is now equivalent to (space) in alpha expressions.

Error Procedure: If TECO should read off the end of the input tape and appear to be waiting for more, a search command has probably failed either because the character string was improperly typed or because the input tape did not actually contain what it was thought to contain. To aid in determining where the failure occured, TECO may be restarted at location 101 (octal). it will then type out the string of characters that it was attempting to find. If additional information is needed, CONTINUE may be pressed, causing TECO to type that portion of its command list following the failed search command.

NEW COMMANDS

/ (slash)

Causes TECO to treat as a comment and ignore all characters between the slash and the first following carriage return. If switch 2 is ON during execution, or if two successive slashes begin the comment , the comment will be typed on line.

s or S

There are now two different modes for the search command. The lower case search works as before, i.e., it searches the current page, punches, and brings in additional pages until satisfied. This will be called the unlimited search. The upper case search (the 's' must be preceded by an upper case and immediately followed by a lower case shift) is called the limited search, and works as follows: The search begins at the current position of the pointer and does not pass beyond the end of the current page. If the designated string is found, the entire 's2 command with its string of characters, up to and including the terminating center dot, stands as the beginning of an alpha expression, having the value -1. The pointer will be following the last character of the string, as before. If the designated string cannot be found on the current page, the pointer comes to rest at the end of the page, and the entire 's' command has the value of zero.

If an argument is given immediately before either the limited or the unlimited search, it will be taken as a position, either of a storage shelf, or of a position in the buffer, which specifies the character string to be found. This string must be constructed in the same manner as the string following the regular search command, that is, it must have at least two characters, one of which is the terminating center dot. Nothing except commands should follow the 's' in this case.

LOOP AND CONDITIONAL

(alpha, [commands]·)

center dot; open paren; alpha expression; comma; arbitrary length string of commands; center dot; close paren

it is now possible to execute the same sequence of commands a number of times controlled by the stored program. The center dot - open paren serves to indicate the beginning of the loop. The alpha expression preceding the comma is evaluated, and if negative, the commands up to the corresponding center dot - close paren are executed. TECO will then loop back to the matching center dot - open paren, re-evaluate the alpha expression, and if negative, again execute the enclosed commands. If on the first or any following evaluations of the alpha expression, a positive quantity is computed, TECO will skip all commands up to the corresponding close paren. Nesting of loops to a depth of 64. is allowed. The conditional is made by causing the enclosed commands to change the alpha expression so that TECO will loop at most once.

Example: Suppose we have a source program written in MACRO in which the pseudo-instruction 'flex' is used, and it is desired to convert this so that it may be assembled with DANTRAN in which the symbol must be 'flexo'. The first example shown will convert all such occurences on a single page, and when nested as in the second, will convert all such occurences on several pages.

```
(Sflex; 1r 1ho )    (-1, sflex ·5r (Sflex ; 1r 1ho))
```

'q' registers: May have any number 0-9, or any letter a-z following
the 'q'. Provides a total of 36. such registers.

If an alpha expression is typed before an 'i' command, it will work in this manner: If the value of the expression is 0≤alpha≤777 (octal), TECO will convert the quantity into octal concise code, eliminate leading zeroes and insert the remaining 1-3 characters into the buffer at the current position of the pointer. If the value of the expression is not within the above limits, TECO will assume that the expression is already concise code, left justified, and will insert from 1 to 3 characters into the buffer, eliminating trailing spaces.

If the 'a' command is preceeded by an alpha expression, the following will occur: If the expression is positive, the entire 'a' command is a quantity which has the value of the alpha characters following the 'a', left justified, with spaces filled in on the right. If the alpha expression is negative, the 'a' command is a quantity which has the value of the -alpha characters after the current position of the pointer in the buffer. The argument must always be -3≤alpha≤+3.

Examples:

```
            3axyzuq0  will put 273031 into register q0
            1axuqf    will put 270000 into register qf

1uq0 (q0-5, q0i q0+1uq0 1o77 ) would put 4 lines
                      into the buffer, beginning with
                      1,2,3,4 respectively.
```

If example 2 above had been given, and q0i were then given, the character 'x' would be inserted.

MACROS

A limited form of macro is available to eliminate writing the same string of commands multiple times.

The command 'im' indicates a macro <u>definition</u>. TECO will take the characters up to the next ',' (comma) as the macro name. The first three only are significant. The string of characters up to '$\bar{.}$' (overbar-period) is then defined as the text of the macro. Thereafter, whenever the previously mentioned name is written as a command, <u>with an overbar over the first character in the name, and a comma following the last character</u>, the previously defined string of characters will be executed.

Example: 1mpage, .uq9 bj ipage ·q0i 1o77 q0+1uq0 q9j $\bar{.}$

A line with the word 'page' followed by a number would appear on the top of each page for which the macro call ' $\bar{p}$age, ' was given, while that page was in the buffer. The number is increased by 1 on each use of the macro.